DRAFT
VAN VLECK CORRECTION FOR THE GBT CORRELATOR

FREDERIC R. SCHWAB

February 11, 2002

ABSTRACT. I have recently reviewed the Glish/AIPS++ Van Vleck correction code and here wish to offer some recommendations on streamlining and improving that code. These recommendations derive from recent work of mine on optimal quantization functions for multi-level digital correlators, with applications to ALMA and EVLA, as well as the GBT. That work is still in progress, and is to be published elsewhere [3].

1. Introduction. I have recently reviewed the AIPS++ Van Vleck correction codes for the GBT, written in Glish and C, and here want to offer some recommendations on streamlining and improving that code. The AIPS++ software is derived from codes developed at Arecibo by Murray Lewis and in Tucson by Andy Dowd for the 12-Meter Telescope. Murray Lewis's approximations to the 9-level Van Vleck correction are based on analytic expressions for the 9-level correlator output that I provided to Mike Davis some years ago (I believe in 1996). Murray Lewis's memoranda [1,2] which furnish these approximations, and provide their derivation, are not referenced in the Glish/AIPS++ code. In the absence of Murray's documentation the GBT code would be, for the most part, inscrutable and unverifiable.

Murray Lewis's first memorandum [1] presents approximations to the 9-level Van Vleck correction curve for the case in which the true correlation, ρ satisfies $|\rho| \leq 0.975$; his second memorandum [2] treats the very high correlation regime, $0.975 \leq |\rho| \leq 1$. His approach is appropriate both for autocorrelation and cross-correlation modes, assuming stationary zero-mean bivariate normal inputs to the correlator, of identical variance σ^2. (For the cross-correlation modes his approximations will be inadequate if the signal input levels to the correlator have not been adjusted to have equal variance.) In the 9-level case, the optimal spacing of the quantizer input levels is $\sim 0.5338222\sigma$, implying that the first positive input threshold value is one-half that value: or $v_1 = v_{\mathrm{opt}} \approx 0.266911\sigma$.

Lewis's memoranda provide not only the Van Vleck correction curves for the case of the optimal threshold setting, $v_1 = v_{\mathrm{opt}}$, but also for suboptimal cases in which the threshold level settings may differ from the optimal setting in discrete steps, by some multiple of 10%—namely, $v_1 = kv_{\mathrm{opt}}$, for $k = 0.3, 0.4, 0.5, \ldots, 2.0$. I believe that his intention was to linearly interpolate the coefficients for intermediate values of the measured thresholds; I have not scrutinized the Glish/AIPS++ code closely enough to see whether that interpolation is performed. In fact, the coefficients for the 9-level Van Vleck correction appear to be derived from work done earlier, or perhaps later, than that described in the available memoranda, as the coefficients in the code do not quite match those in the memoranda.

The 3-level correction in the Glish/AIPS++ code employs polynomial approximations which no doubt are sufficiently accurate, but no reference to the source of these approximations is given in the code.

Really, I think that anyone who might look at the Glish/AIPS++ code would agree, although it might indeed be working as intended, and acceptably well, that at least for aesthetic reasons it ought to be replaced by a cleaner code. So—rather than dissect the code any further—I simply intend, in the next few sections, to describe how I would replace it.

2. Computation of Van Vleck Correction Curves. Because other multi-level digital correlators besides the GBT correlator are under development (e.g., for ALMA and EVLA), I decided to write a general-purpose Van Vleck code that could be used for those applications as well. Whereas the GBT correlator operates in 3-level by 3-level (3×3) and 9×9 modes, the ALMA correlator will operate in 2-bit and 4-bit (4- and 16-level modes) and the EVLA correlator in 4-bit and higher-level modes. My code, given in Appendix A and described below, is general enough to be used for the ALMA and EVLA applications—and, indeed, for most other radio-astronomical correlators, existing or planned.

The idea behind the code in Appendix A is simply to compute, to high accuracy, a sufficient number of points along the Van Vleck curve that fitting a polynomial spline interpolant to these points will yield a sufficiently accurate correction curve. The Van Vleck curves are quite linear for small $|\rho|$, but typically have a hook-like behavior for larger $|\rho| \approx 1$. Therefore, I have chosen to sample with abscissae located at the so-called expanded Chebyshev points [4, p. 27], which are more highly clustered near $\rho = \pm 1$. Somewhat arbitrarily, I have chosen to interpolate among 65 points. This is probably overkill: for the 3-level and 9-level cases—with equi-spaced quantizer input level thresholds, fixed and equi-spaced output levels, but varying input threshold spacing—and employing cubic natural spline interpolation [4], the resulting maximum relative error is less than 10^{-9}. Accurate computation of a small number of points along the Van Vleck curve should easily be affordable in the GBT application, and—once (per correlator dump, typically) having done so—correction of all the lag data (typically many thousands of lag correlation measurements per dump) should be accomplished very quickly, because spline interpolation is very inexpensive (involving just a table-look up, to find the correct set of coefficients, followed by evaluation of a low-order (cubic) polynomial).

The code in Appendix A is completely general, in the sense that the two, x- and y-inputs to the correlator may be quantized to different numbers of levels; the quantizer input voltage thresholds may differ arbitrarily; and the quantizer output levels may be specified arbitrarily. Of course, for the GBT correlator we do not need all this generality—but it comes at almost no added expense.

An n-level quantization function $q(x)$ may be specified by a list V of input voltage thresholds, $-\infty \equiv v_0 < v_1 < v_2 < \cdots < v_{n-1} < v_n \equiv \infty$; and a list W of quantizer output levels, $W = \{w_1, w_2, \ldots, w_n\}$; such that $q(x) = w_k$, whenever x lies between v_{k-1} and v_k.

Suppose the (real-valued) inputs to an m-level by n-level correlator to be drawn from a bivariate normal distribution with means μ_x and μ_y, standard deviations σ_x and σ_y, and correlation coefficient ρ; and thus characterized by the joint probability density function

$$p(x, y, \rho) = \frac{1}{\sigma_x \sigma_y}\, g\left(\frac{x-\mu_x}{\sigma_x},\, \frac{y-\mu_y}{\sigma_y},\, \rho\right), \tag{1}$$

where

$$g(x, y, \rho) \equiv \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}\left(\frac{x^2 - 2\rho x y + y^2}{1-\rho^2}\right)\right) \tag{2}$$

is the standard bivariate normal probability function [5, p. 936]. And suppose that the (finite) voltage input thresholds of the x-quantizer are $\{v_{x,1}, v_{x,2}, \ldots, v_{x,m-1}\}$ and that the corresponding output levels are $\{w_{x,1}, w_{x,2}, \ldots, w_{x,m}\}$; and for the y-quantizer that the finite thresholds and the output levels are $\{v_{y,1}, v_{y,2}, \ldots, v_{y,n-1}\}$ and $\{w_{y,1}, w_{y,2}, \ldots, w_{y,n}\}$, respectively. Then, by integration of the joint probability density function (Eq. 1), the expectation value of the correlator output is

$$r(\rho) = \frac{1}{\sigma_x \sigma_y} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{x,i} w_{y,j} \int_{v_{x,i-1}}^{v_{x,i}} \int_{v_{y,j-1}}^{v_{y,j}} g\left(\frac{x-\mu_x}{\sigma_x},\, \frac{y-\mu_y}{\sigma_y},\, \rho\right) dy\, dx\,. \tag{3}$$

And (by the Second Fundamental Theorem of Calculus) the derivative, with respect to ρ, of the expectation value of the correlator output is given simply by

$$h(\rho) \equiv \frac{dr}{d\rho} = \frac{1}{\sigma_x \sigma_y} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (w_{x,i+1} - w_{x,i})(w_{y,j+1} - w_{y,j})\, g\left(\frac{v_{x,i}-\mu_x}{\sigma_x},\, \frac{v_{y,j}-\mu_y}{\sigma_y},\, \rho\right). \tag{4}$$

(This result would also obtain by application of Price's Theorem [6], [7].) The expected correlator output thus can be expressed not only as in Equation 3, but also as

$$r(\rho) = r(0) + \int_0^{\rho} h(\rho')\, d\rho'\,. \tag{5}$$

Hence $r(\rho)$, for any ρ, may be computed to any desired accuracy via numerical quadrature. Observe that $r(0)$ is equal to zero whenever the quantization functions possess odd symmetry and $\mu_x = \mu_y = 0$, but otherwise (from Eq. 3, after simplification)

$$r(0) = \frac{1}{4}\left(\sum_{i=1}^{m} w_{x,i}\left(\operatorname{erf}\left(\frac{\mu_x - v_{x,i}}{\sigma_x\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\mu_x - v_{x,i-1}}{\sigma_x\sqrt{2}}\right)\right)\right) \times$$
$$\left(\sum_{j=1}^{n} w_{y,j}\left(\operatorname{erf}\left(\frac{\mu_y - v_{y,j}}{\sigma_y\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\mu_y - v_{y,j-1}}{\sigma_y\sqrt{2}}\right)\right)\right). \tag{6}$$

In the code in Appendix A, I use the automatic quadrature subroutine `dqags` of the Fortran package `QUADPACK` [8] to perform the numerical integration. The automated quadrature routine selects suitable Gaussian quadrature formulae, and appropriate sampling abscissae, to achieve user-specified absolute and relative error bounds. Another quadrature routine could be substituted, if desired, in the Glish/AIPS++ adaptation of this code.

The end result of the code in Appendix A is a list of ordered pairs $\{(r_i, \rho_i) \mid i = 1, \ldots, N\}$, with $-1 \equiv \rho_1 < \rho_2 < \cdots < \rho_N \equiv 1$ (a set of expanded Chebyshev points, as mentioned above, coarsely spaced) and satisfying $r(\rho_i) = r_i$, which is suitable for recovering true correlation ρ for any measured correlation r, by spline interpolation. Since I believe Glish/AIPS++ already contains a suitable spline code, I have not included such a code in Appendix A.

3. Alternative Computation of the Van Vleck Correction Curves, via the Bivariate Normal Integral. Each of the double integrals occurring within the double summation in Equation 3 may be expressed in terms of the bivariate normal integral [5, p. 936],

$$L(h,k,\rho) \equiv \int_h^\infty \int_k^\infty g(x,y,\rho)\,dy\,dx\,. \tag{7}$$

Specifically,

$$\int_a^b \int_c^d g(x,y,\rho)\,dy\,dx = L(a,c,\rho) - L(a,d,\rho) - L(b,c,\rho) + L(b,d,\rho)\,. \tag{8}$$

Expressions $L(h,k,\rho)$ with h or k equal to $-\infty$ simplify to become error functions, and those with h or k equal to $+\infty$ reduce to zero. If quantizer input level thresholds are spaced symmetrically about the origin, then some terms can be combined. I have a Mathematica code which automatically generates and simplifies the formula for $r(\rho)$, for any desired quantization scheme, but I do not have such general a code written in Fortran or C.

In Appendix B, I do, however, provide a Fortran code for computation of the 3-level $\times$ 3-level and 9-level $\times$ 9-level Van Vleck curves in terms of the bivariate normal integral, for the special case of equi-spaced input thresholds and equi-spaced output levels. This code may be appropriate for the GBT, especially if it should happen that the code of Appendix A is too slow.

I use the subroutine `BVND`, written by Alan Genz of the Department of Mathematics at Washington State University, and based on the algorithms given in [9], [10], for computation of the bivariate normal integrals.

This code (like the code of Appendix A) would be appropriate for correcting both autocorrelation and cross-correlation data. I allow for the possibility of inputs with non-zero mean (i.e., possible "DC offsets"), because Rick Fisher (private communication) has suggested that this may be necessary in the case of the GBT correlator.

4. Computing Quantizer Threshold Spacing from the Measured Zero-Lag Autocorrelation. In Appendix C, I give a Fortran code which can be used, in the case of quantization functions with equi-spaced input threshold and equi-spaced output levels, to infer the quantizer input threshold spacing from the measured zero-lag autocorrelation (assuming bivariate normal inputs to the correlator, with zero mean). Given the number of quantization levels and the measured zero-lag autocorrelation, the code returns the inferred value of the first positive input threshold, v_1, (in units of the r.m.s. input level).

For the case of odd n, the expectation of the zero-lag autocorrelation is

$$r(1) = \sum_{k=1}^{(n-1)/2} (2k-1)\,\mathrm{erfc}\left(\frac{(2k-1)v_1}{\sqrt{2}}\right) , \tag{9}$$

and for even n

$$r(1) = 1 + 8 \sum_{k=1}^{(n-2)/2} k \ \mathrm{erfc}\left(\frac{kv_1}{\sqrt{2}}\right) . \tag{10}$$

The code of Appendix C solves for v_1 by means of Newton's method. I have tested it thoroughly for the cases $n = 3, 4, 5, \ldots, 16$, but not for larger values of n. For typical values of the zero lag, convergence is achieved in 5–6 iterations. For pathologically extreme values the iteration limit would need to be increased.

For the $n = 3$ case, $v_1 = \sqrt{2}\,\mathrm{erfc}^{-1}(r(1))$, where erfc^{-1} is the inverse of the error function. For this case, the inverse error function code of Appendix E could be used instead.

5. Deriving 3-Level or 9-Level Quantizer Thresholds from Quantizer Zone Population Counts. In the case of possible DC offsets in the inputs to the GBT correlator, it may be useful to try to infer the quantizer input threshold spacings and the mean signal levels from the correlator "self counts"—i.e., from the population counts at each of the quantization levels. The problem is that these counts are not provided for the full duration of an integration period, but only for, I believe, a 20-msec interval at the start of (or preceding?) the integration period. So the statistics might really be inadequate. When operating in 9-level mode, only the counts for the lowermost three levels, middle three levels, and uppermost three levels are provided.

From the population counts it is straightforward to derive estimates of the mean and r.m.s. input signal levels. A template code for this purpose, written for Mathematica, is given in Appendix D. It requires a code for the inverse of the error function; a suitable template code is given in Appendix E. This code is based on approximations published by Blair, Edwards, and Johnson [11]. From their tabulations of rational approximations to erf^{-1} and erfc^{-1}, I chose approximations suitable to achieve maximum relative errors less than 10^{-7} over the entire domain of definition of the functions. (The older Van Vleck code uses an approximation to $\mathrm{erf}^{-1}(x)$, taken from [12], which is valid only over the interval $|x| \leq 0.75$.)

6. Discussion. I believe that here I've provided most of the pieces which would be necessary for a clean, reasonably well-documented Van Vleck code for the GBT. Further programming effort would be needed to coordinate the logic flow in calling up these pieces and correcting the actual data. If the approach of Appendix A is undertaken, one might want to re-organize the steps somewhat, depending on what degree of modularization, etc., is desired. As I'm not a competent C or C++ programmer, I've provided only template codes, written in either Fortran or the programming language of Mathematica.

My code should be fully adequate for the case of cross-correlation data. In this case, one should first process the two sets of auto-correlation data, in order to infer (from either the zero-lag auto-correlation, or the population counts) the relation of input signal statistics to quantizer characteristics. This code would also be applicable to 3-level $\times$ 9-level operation of the correlator, but I believe that mode of operation is not under consideration.

References

[1] B. Murray Lewis, "r-to-ρ translations for the nine-level correlator", internal memorandum addressed to Phil Perrilat, National Astronomy and Ionosphere Center, Feb. 14, 1997; available at `http://www.naic.edu/~astro/general_info/correlator`.

[2] B. Murray Lewis, "r-to-ρ translations for the nine-level correlator II: for the cases of very high correlation values, in lags other than the zero lag", internal memorandum addressed to Phil Perrilat, National Astronomy and Ionosphere Center, Apr. 29, 1997; available at `http://www.naic.edu/~astro/general_info/correlator`.

[3] Frederic R. Schwab, "Optimal quantization functions for multi-level digital correlators", NRAO, in preparation.

[4] Carl de Boor, *A Practical Guide to Splines*, Springer–Verlag, New York, 1978.

[5] Milton Abramowitz and Irene A. Stegun, Eds., *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards Applied Mathematics Series, **55**, U.S. Government Printing Office, Washington, D.C., 1970; reprint Dover, New York, 1974.

[6] Robert Price, "A useful theorem for nonlinear devices having Gaussian inputs", *IRE Trans. on Information Theory*, **4** (1958) 69–72.

[7] Jon B. Hagen and Donald T. Farley, "Digital-correlation techniques in radio science", *Radio Science*, **8** (1973) 775–784.

[8] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner, *QUADPACK: A Subroutine Package for Automatic Integration*, Springer–Verlag, Berlin, 1983; this public-domain Fortran package is available on the Web at `http://www.netlib.org`.

[9] Zvi Drezner, "Computation of the bivariate normal integral", *Mathematics of Computation*, **32** (1978) 277–279.

[10] Zvi Drezner and George O. Wesolowsky, "On the computation of the bivariate normal integral", *J. Statistical Computation and Simulation*, **35** (1989) 101–107.

[11] J. M. Blair, C. A. Edwards, and J. H. Johnson, "Rational Chebyshev approximations for the inverse of the error function", *Mathematics of Computation*, **30** (1976) 827–830 + microfiche appendix.

[12] Frederic R. Schwab, "Quantization correction of VLA correlation measurements", VLA Computer Memo. No. 150, NRAO, Socorro, Dec. 1978 (released Oct. 1979).

Appendix A. A Fortran Code to Compute Van Vleck Correction Curves

This Van Vleck correction code is fully documented in Section 2, above, and in the comment lines within the code. (Except that quantizer input threshold levels here are given in units of the r.m.s. signal input voltage levels, σ_x and σ_y.) I have here omitted the `QUADPACK` code, since it is readily available on the Web (i.e., from Netlib).

```
      module quantizerdeclarations
      integer, parameter :: nmax=256
      integer :: nx,ny
      double precision :: qx(2,nmax),qy(2,nmax)
      end module

      program testvv
c
c  Program to compute highly accurate Van Vleck correction curves
c  for an m-level x n-level digital correlator, for arbitrary m and n,
c  and arbitrarily specified quantization input thresholds and output
c  levels, for bivariate normal inputs possibly with non-zero means
c  (i.e., allowing d.c. offsets)
c
c  What we'll produce is a highly accurate table of ordered pairs
c  {( r(rho(i)), rho(i) ), i=1,...,npts}, with rho(i) - the true
c  correlation in the range [-1,1] - and r(rho(i)) the expectation
c  of the (unnormalized) correlator output when the true correlation is
c  equal to rho(i).  Then the idea would be to fit a spline approximation
c  to these tabular data and apply that correction to the measured
c  correlations.  Choosing 65 points appropriately spaced (i.e., much
c  more densely spaced for large |rho|) suffices in achieving a relative
c  accuracy of approximately 10^-7 for the 9-level x 9-level case if
c  cubic natural spline interpolation is employed.  (65 points is
c  probably overkill - perhaps one-half, or even one-quarter, that
c  number of points would suffice, but I haven't checked yet.)
c
      use quantizerdeclarations
      integer, parameter :: npts=65
      double precision :: v1,v1x,v1y,mux,muy,rs(npts),rhos(npts)

c  First set up the definition of the quantization functions as in
c  one of the examples below:

c  Optimal 3-level by 3-level quantization function:
c  (I.e., the 3-level by 3-level quantization function for
c     zero-mean input signals with voltage thresholds set
c     at the optimal values of +/- ~0.612003 sigma.)
c
      nx=3
      v1=.61200318096d0
      qx(1,1:nx-1)=(/ -v1,v1 /)
      qx(2,1:nx)=(/ -1d0,0d0,1d0 /)
      ny=3
      qy(1,1:ny-1)=(/ -v1,v1 /)
      qy(2,1:ny)=(/ -1d0,0d0,1d0 /)
```

```
c  Like the above, but if the threshold for the x-quantizer
c     were set non-optimally at 0.6 sigma and there were a d.c.
c     offset of 0.01 sigma in the x-inputs, and if the y-quantizer
c     were set non-optimally at 0.7 sigma and there were a d.c.
c     offset of -.02 sigma in the y-inputs
c
      nx=3
      v1x=.6d0
      mux=.01d0
      qx(1,1:nx-1)=(/ -v1x,v1x /)-mux
      qx(2,1:nx)=(/ -1d0,0d0,1d0 /)
      ny=3
      v1y=.7d0
      muy=-.02d0
      qy(1,1:ny-1)=(/ -v1y,v1y /)-muy
      qy(2,1:ny)=(/ -1d0,0d0,1d0 /)

c  Optimal 3-level by 9-level quantization function:
c  (I.e., no d.c. offsets, and thresholds set optimally for both
c     the 3-level x-quantizer's input signal and the 9-level
c     y-quantizer's input signal.)
c
      nx=3
      v1=.61200318096d0
      qx(1,1:nx-1)=(/ -v1,v1 /)
      qx(2,1:nx)=(/ -1d0,0d0,1d0 /)
      ny=9
      v1=.26691110435d0
      qy(1,1:ny-1)=(/ -7*v1,-5*v1,-3*v1,-v1,v1,3*v1,5*v1,7*v1 /)
      qy(2,1:ny)=(/ -4d0,-3d0,-2d0,-1d0,0d0,1d0,2d0,3d0,4d0 /)

c  Optimal 9-level by 9-level quantization function:
c  (I.e., the 9-level by 9-level quantization function for
c     zero-mean input signals with voltage thresholds set
c     at the optimal values of +/- (2k-1)*0.266911 sigma, k=1,2,3,4.)
c
      nx=9
      v1=.26691110435d0
      qx(1,1:nx-1)=(/ -7*v1,-5*v1,-3*v1,-v1,v1,3*v1,5*v1,7*v1 /)
      qx(2,1:nx)=(/ -4d0,-3d0,-2d0,-1d0,0d0,1d0,2d0,3d0,4d0 /)
      ny=9
      qy(1,1:ny-1)=(/ -7*v1,-5*v1,-3*v1,-v1,v1,3*v1,5*v1,7*v1 /)
      qy(2,1:ny)=(/ -4d0,-3d0,-2d0,-1d0,0d0,1d0,2d0,3d0,4d0 /)

c  Now call the subroutine vanvleck to generate the table of
c  rs and rhos appropriate for spline interpolation to get rho(r)
c  for any observed (unnormalized) correlator output measurement, r:
c
      call vanvleck(npts,rs,rhos)
      stop
      end

      subroutine vanvleck(npts,rs,rhos)
      integer npts
      double precision, parameter :: pi=3.1415926535897932d0
```

```
      double precision :: rhos(npts),rs(npts),rapprox,rzero,r0
      integer :: i
c  Get r(0), which has to be added to the integrated derivative
c  of dr/drho, to get r(rho).
      rzero=r0()
      do i=1,npts
c  For the rhos, choose the expanded Chebyshev points:
         rhos(i)=-cos((2*i-1)*pi/(2*npts))/cos(pi/(2*npts))
c  Now call the function rapprox and add r(0), to get a highly accurate
c  value of the expected correlator output for that value of rho:
         rs(i)=rzero+rapprox(rhos(i))
         print *,i,rs(i),rhos(i)
      end do
      end

      double precision function drbydrho(rho)
c  For given rho, and given quantization functions, this function
c  subroutine computes - via Price's theorem - the value dr/drho of
c  the derivative, with respect to rho, of the expected value of the
c  correlator output.  (Another function, rapprox, then will numerically
c  integrate from 0 to rho, to get r(rho)-r(0) ).
      use quantizerdeclarations
      double precision :: rho,g,x,y,s
      double precision, parameter :: twopi=6.2831853071795865d0
      integer :: i,j
      g(x,y,rho)=exp(-.5d0*(x**2-2d0*rho*x*y+y**2)/(1d0-rho**2))/
     & (twopi*sqrt(1d0-rho**2))
      s=0d0
      do i=1,nx-1
         do j=1,ny-1
            s=s+(qx(2,i+1)-qx(2,i))*(qy(2,j+1)-qy(2,j))*
     &          g(qx(1,i),qy(1,j),rho)
            end do
         end do
      drbydrho=s
      end function

      double precision function rapprox(rho)
c  For given rho, this function subroutine produces a high-accuracy
c  numerical approximation to the expected value of the correlator
c  output r(rho)-r(0), by numerical integration of dr/drho.  It calls
c  the standard QUADPACK adaptive Gaussian quadrature procedure, dqags,
c  to do the numerical integration.
      double precision :: rho,drbydrho,a,b,epsabs,epsrel,result,abserr
      double precision :: work(4096)
      integer :: neval,ier,limit,lenw,last,iwork(1024)
      external drbydrho
      a=0d0
      b=rho
      epsabs=1d-12
      epsrel=1d-12
      limit=1024
      lenw=4*limit
      call dqags(drbydrho,a,b,epsabs,epsrel,result,abserr,neval,ier,
     &   limit,lenw,last,iwork,work)
```

```
c      print *,neval
      if (ier.ne.0) print *,"Error in dqags, ier=",ier
      rapprox=result
      return
      end

      double precision function r0()
c  This function computes r(0) - the expectation of the correlator
c  output when the true correlation rho=0.  Note that r(0)=0 whenever
c  the quantization functions possess odd symmetry and the mean signal
c  input levels are zero.  Otherwise, in general, r(0).ne.0.
      use quantizerdeclarations
      double precision :: rt2,sx,sy
      integer i,j
      rt2=sqrt(2d0)
      sx=qx(2,1)*(erf(qx(1,1)/rt2)+1d0)
      if (nx.gt.2) then
         do i=2,nx-1
            sx=sx+qx(2,i)*(erf(qx(1,i)/rt2)-erf(qx(1,i-1)/rt2))
         end do
      end if
      sx=sx+qx(2,nx)*(1d0-erf(qx(1,nx-1)/rt2))
      sy=qy(2,1)*(erf(qy(1,1)/rt2)+1d0)
      if (ny.gt.2) then
         do j=2,ny-1
            sy=sy+qy(2,j)*(erf(qy(1,j)/rt2)-erf(qy(1,j-1)/rt2))
         end do
      end if
      sy=sy+qy(2,ny)*(1d0-erf(qy(1,ny-1)/rt2))
      r0=.25d0*sx*sy
      return
      end


c***********************************************************************
c This is the end of the custom code.
c***********************************************************************
c All the code that follows is standard mathematical software library
c code, namely, the relevant QUADPACK code.  Glish/AIPS++ probably have
c a suitable quadrature routine that could be substituted for dqags.
c Similarly, Glish/AIPS++ already have suitable cubic natural spline
c interpolation codes, so I haven't supplied them
c
c                                              - F. Schwab, Dec. 3, 2001
c***********************************************************************
```

Appendix B. Fortran Code for Approximation of the 3-Level and 9-Level Van Vleck Curves in Terms of the Bivariate Normal Integral

As described in Section 3, the correlator response $r(\rho)$ may be expressed in terms of the bivariate normal integral. The two Fortran subroutines, r3 and r9, listings of which appear below, may be used to approximate the 3-level × 3-level and 9-level × 9-level Van Vleck curves, for the special case of equi-spaced quantizer input voltage thresholds and equi-space output levels. Here, the mean input levels (mux and muy) and the first positive input thresholds (v1x and v1y) are assumed to be given in units of the respective r.m.s. input levels μ_x and μ_y.

The subroutine BVND, written by Alan Genz of the Department of Mathematics at Washington State University, and based on the algorithms given in [9], [10], is used for computation of the bivariate normal integrals. This code may appropriate for use with the GBT, especially if it should happen that the code of Appendix A is too slow for that application.

```
      double precision function r3(mux,muy,v1x,v1y,rho)
      double precision mux,muy,v1x,v1y,rho,L,h,k,r,bvnd,rt2
      L(h,k,r)=bvnd(h,k,r)
      rt2=sqrt(2d0)
      r3=.5d0*(erf((-mux+v1x)/rt2)-erf((mux+v1x)/rt2)+
     & erf((-muy+v1y)/rt2)-erf((muy+v1y)/rt2))+
     & L(-mux-v1x,-muy-v1y,rho)+L(-mux-v1x,-muy+v1y,rho)+
     & L(-mux+v1x,-muy-v1y,rho)+L(-mux+v1x,-muy+v1y,rho)-1d0
      return
      end
```

```
      double precision function r9(mux,muy,v1x,v1y,rho)
      double precision mux,muy,v1x,v1y,rho,L,h,k,r,bvnd,rt2
      L(h,k,r)=bvnd(h,k,r)
      rt2=sqrt(2d0)
      r9=-16d0+2d0*(-erf((mux-7*v1x)/rt2)-erf((mux-5*v1x)/rt2)-
     & erf((mux-3*v1x)/rt2)+erf((-mux+v1x)/rt2)-
     & erf((mux+v1x)/rt2)-erf((mux+3*v1x)/rt2)-
     & erf((mux+5*v1x)/rt2)-erf((mux+7*v1x)/rt2)-
     & erf((muy-7*v1y)/rt2)-erf((muy-5*v1y)/rt2)-
     & erf((muy-3*v1y)/rt2)+erf((-muy+v1y)/rt2)-
     & erf((muy+v1y)/rt2)-erf((muy+3*v1y)/rt2)-
     & erf((muy+5*v1y)/rt2)-erf((muy+7*v1y)/rt2))+
     & L(-mux-7*v1x,-muy-7*v1y,rho)+L(-mux-7*v1x,-muy-5*v1y,rho)+
     & L(-mux-7*v1x,-muy-3*v1y,rho)+L(-mux-7*v1x,-muy-v1y,rho)+
     & L(-mux-7*v1x,-muy+v1y,rho)+L(-mux-7*v1x,-muy+3*v1y,rho)+
     & L(-mux-7*v1x,-muy+5*v1y,rho)+L(-mux-7*v1x,-muy+7*v1y,rho)+
     & L(-mux-5*v1x,-muy-7*v1y,rho)+L(-mux-5*v1x,-muy-5*v1y,rho)+
     & L(-mux-5*v1x,-muy-3*v1y,rho)+L(-mux-5*v1x,-muy-v1y,rho)+
     & L(-mux-5*v1x,-muy+v1y,rho)+L(-mux-5*v1x,-muy+3*v1y,rho)+
     & L(-mux-5*v1x,-muy+5*v1y,rho)+L(-mux-5*v1x,-muy+7*v1y,rho)+
     & L(-mux-3*v1x,-muy-7*v1y,rho)+L(-mux-3*v1x,-muy-5*v1y,rho)+
     & L(-mux-3*v1x,-muy-3*v1y,rho)+L(-mux-3*v1x,-muy-v1y,rho)+
     & L(-mux-3*v1x,-muy+v1y,rho)+L(-mux-3*v1x,-muy+3*v1y,rho)+
     & L(-mux-3*v1x,-muy+5*v1y,rho)+L(-mux-3*v1x,-muy+7*v1y,rho)+
     & L(-mux-v1x,-muy-7*v1y,rho)+L(-mux-v1x,-muy-5*v1y,rho)+
     & L(-mux-v1x,-muy-3*v1y,rho)+L(-mux-v1x,-muy-v1y,rho)+
     & L(-mux-v1x,-muy+v1y,rho)+L(-mux-v1x,-muy+3*v1y,rho)+
     & L(-mux-v1x,-muy+5*v1y,rho)+L(-mux-v1x,-muy+7*v1y,rho)+
```

```
     & L(-mux+v1x,-muy-7*v1y,rho)+L(-mux+v1x,-muy-5*v1y,rho)+
     & L(-mux+v1x,-muy-3*v1y,rho)+L(-mux+v1x,-muy-v1y,rho)+
     & L(-mux+v1x,-muy+v1y,rho)+L(-mux+v1x,-muy+3*v1y,rho)+
     & L(-mux+v1x,-muy+5*v1y,rho)+L(-mux+v1x,-muy+7*v1y,rho)+
     & L(-mux+3*v1x,-muy-7*v1y,rho)+L(-mux+3*v1x,-muy-5*v1y,rho)+
     & L(-mux+3*v1x,-muy-3*v1y,rho)+L(-mux+3*v1x,-muy-v1y,rho)+
     & L(-mux+3*v1x,-muy+v1y,rho)+L(-mux+3*v1x,-muy+3*v1y,rho)+
     & L(-mux+3*v1x,-muy+5*v1y,rho)+L(-mux+3*v1x,-muy+7*v1y,rho)+
     & L(-mux+5*v1x,-muy-7*v1y,rho)+L(-mux+5*v1x,-muy-5*v1y,rho)+
     & L(-mux+5*v1x,-muy-3*v1y,rho)+L(-mux+5*v1x,-muy-v1y,rho)+
     & L(-mux+5*v1x,-muy+v1y,rho)+L(-mux+5*v1x,-muy+3*v1y,rho)+
     & L(-mux+5*v1x,-muy+5*v1y,rho)+L(-mux+5*v1x,-muy+7*v1y,rho)+
     & L(-mux+7*v1x,-muy-7*v1y,rho)+L(-mux+7*v1x,-muy-5*v1y,rho)+
     & L(-mux+7*v1x,-muy-3*v1y,rho)+L(-mux+7*v1x,-muy-v1y,rho)+
     & L(-mux+7*v1x,-muy+v1y,rho)+L(-mux+7*v1x,-muy+3*v1y,rho)+
     & L(-mux+7*v1x,-muy+5*v1y,rho)+L(-mux+7*v1x,-muy+7*v1y,rho)
      return
      end


      DOUBLE PRECISION FUNCTION BVND( DH, DK, R )
*
*     A function for computing bivariate normal probabilities.
*
*       Alan Genz
*       Department of Mathematics
*       Washington State University
*       Pullman, WA 99164-3113
*       Email : alangenz<place an "at" symbol here>wsu.edu
*
*    This function is based on the method described by
*        Drezner, Z and G.O. Wesolowsky, (1989),
*        On the computation of the bivariate normal inegral,
*        Journal of Statist. Comput. Simul. 35, pp. 101-107,
*    with major modifications for double precision, and for |R| close to 1.
*
* BVND - calculate the probability that X is larger than DH and Y is
*       larger than DK.
*
* Parameters
*
*   DH  DOUBLE PRECISION, integration limit
*   DK  DOUBLE PRECISION, integration limit
*   R   DOUBLE PRECISION, correlation coefficient
*
      DOUBLE PRECISION DH, DK, R, ZERO, TWOPI
      INTEGER I, IS, LG, NG
      PARAMETER ( ZERO = 0, TWOPI = 6.283185307179586D0 )
      DOUBLE PRECISION X(10,3), W(10,3), AS, A, B, C, D, RS, XS, BVN
      DOUBLE PRECISION PHID, SN, ASR, H, K, BS, HS, HK
*     Gauss Legendre Points and Weights, N =  6
      DATA ( W(I,1), X(I,1), I = 1,3) /
     &  0.1713244923791705D+00,-0.9324695142031522D+00,
     &  0.3607615730481384D+00,-0.6612093864662647D+00,
     &  0.4679139345726904D+00,-0.2386191860831970D+00/
```

```
*     Gauss Legendre Points and Weights, N = 12
      DATA ( W(I,2), X(I,2), I = 1,6) /
     &  0.4717533638651177D-01,-0.9815606342467191D+00,
     &  0.1069393259953183D+00,-0.9041172563704750D+00,
     &  0.1600783285433464D+00,-0.7699026741943050D+00,
     &  0.2031674267230659D+00,-0.5873179542866171D+00,
     &  0.2334925365383547D+00,-0.3678314989981802D+00,
     &  0.2491470458134029D+00,-0.1252334085114692D+00/
*     Gauss Legendre Points and Weights, N = 20
      DATA ( W(I,3), X(I,3), I = 1, 10 ) /
     &  0.1761400713915212D-01,-0.9931285991850949D+00,
     &  0.4060142980038694D-01,-0.9639719272779138D+00,
     &  0.6267204833410906D-01,-0.9122344282513259D+00,
     &  0.8327674157670475D-01,-0.8391169718222188D+00,
     &  0.1019301198172404D+00,-0.7463319064601508D+00,
     &  0.1181945319615184D+00,-0.6360536807265150D+00,
     &  0.1316886384491766D+00,-0.5108670019508271D+00,
     &  0.1420961093183821D+00,-0.3737060887154196D+00,
     &  0.1491729864726037D+00,-0.2277858511416451D+00,
     &  0.1527533871307259D+00,-0.7652652113349733D-01/
      SAVE X, W
      IF ( ABS(R) .LT. 0.3D0 ) THEN
         NG = 1
         LG = 3
      ELSE IF ( ABS(R) .LT. 0.75D0 ) THEN
         NG = 2
         LG = 6
      ELSE
         NG = 3
         LG = 10
      ENDIF
      H = DH
      K = DK
      HK = H*K
      BVN = 0
      IF ( ABS(R) .LT. 0.925D0 ) THEN
         HS = ( H*H + K*K )/2
         ASR = ASIN(R)
         DO I = 1, LG
            DO IS = -1, 1, 2
               SN = SIN( ASR*(  IS*X(I,NG) + 1 )/2 )
               BVN = BVN + W(I,NG)*EXP( ( SN*HK - HS )/( 1 - SN*SN ) )
            END DO
         END DO
         BVN = BVN*ASR/( 2*TWOPI ) + PHID(-H)*PHID(-K)
      ELSE
         IF ( R .LT. 0 ) THEN
            K = -K
            HK = -HK
         ENDIF
         IF ( ABS(R) .LT. 1 ) THEN
            AS = ( 1 - R )*( 1 + R )
            A = SQRT(AS)
            BS = ( H - K )**2
            C = ( 4 - HK )/8
```

```
         D = ( 12 - HK )/16
         ASR = -( BS/AS + HK )/2
         IF ( ASR .GT. -100 ) BVN = A*EXP(ASR)
     &             *( 1 - C*( BS - AS )*( 1 - D*BS/5 )/3 + C*D*AS*AS/5 )
         IF ( -HK .LT. 100 ) THEN
            B = SQRT(BS)
            BVN = BVN - EXP( -HK/2 )*SQRT(TWOPI)*PHID(-B/A)*B
     &                    *( 1 - C*BS*( 1 - D*BS/5 )/3 )
         ENDIF
         A = A/2
         DO I = 1, LG
            DO IS = -1, 1, 2
               XS = ( A*(  IS*X(I,NG) + 1 ) )**2
               RS = SQRT( 1 - XS )
               ASR = -( BS/XS + HK )/2
               IF ( ASR .GT. -100 ) THEN
                  BVN = BVN + A*W(I,NG)*EXP( ASR )
     &                    *( EXP( -HK*( 1 - RS )/( 2*( 1 + RS ) ) )/RS
     &                    - ( 1 + C*XS*( 1 + D*XS ) ) )
               END IF
            END DO
         END DO
         BVN = -BVN/TWOPI
      ENDIF
      IF ( R .GT. 0 ) BVN =  BVN + PHID( -MAX( H, K ) )
      IF ( R .LT. 0 ) BVN = -BVN + MAX( ZERO, PHID(-H) - PHID(-K) )
   ENDIF
   BVND = BVN
   END

   DOUBLE PRECISION FUNCTION PHID(Z)
*
*     Normal distribution probabilities accurate to 1.e-15.
*     Z = no. of standard deviations from the mean.
*
*     Based upon algorithm 5666 for the error function, from:
*     Hart, J.F. et al, 'Computer Approximations', Wiley 1968
*
*     Programmer: Alan Miller
*
*     Latest revision - 30 March 1986
*
      DOUBLE PRECISION P0, P1, P2, P3, P4, P5, P6,
     &     Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7,
     &     Z, P, EXPNTL, CUTOFF, ROOTPI, ZABS
      PARAMETER(
     &     P0 = 220.20 68679 12376 1D0,
     &     P1 = 221.21 35961 69931 1D0,
     &     P2 = 112.07 92914 97870 9D0,
     &     P3 = 33.912 86607 83830 0D0,
     &     P4 = 6.3739 62203 53165 0D0,
     &     P5 = .70038 30644 43688 1D0,
     &     P6 = .035262 49659 98910 9D0 )
      PARAMETER(
     &     Q0 = 440.41 37358 24752 2D0,
```

```
     &     Q1 = 793.82 65125 19948 4D0,
     &     Q2 = 637.33 36333 78831 1D0,
     &     Q3 = 296.56 42487 79673 7D0,
     &     Q4 = 86.780 73220 29460 8D0,
     &     Q5 = 16.064 17757 92069 5D0,
     &     Q6 = 1.7556 67163 18264 2D0,
     &     Q7 = .088388 34764 83184 4D0 )
      PARAMETER( ROOTPI = 2.5066 28274 63100 1D0 )
      PARAMETER( CUTOFF = 7.0710 67811 86547 5D0 )
*
      ZABS = ABS(Z)
*
*     |Z| &gt; 37
*
      IF ( ZABS .GT. 37 ) THEN
         P = 0
      ELSE
*
*     |Z| &lt;= 37
*
         EXPNTL = EXP( -ZABS**2/2 )
*
*     |Z| &lt; CUTOFF = 10/SQRT(2)
*
         IF ( ZABS .LT. CUTOFF ) THEN
            P = EXPNTL*( ( ( ( ( ( P6*ZABS + P5 )*ZABS + P4 )*ZABS
     &            + P3 )*ZABS + P2 )*ZABS + P1 )*ZABS + P0 )
     &           /( ( ( ( ( ( ( Q7*ZABS + Q6 )*ZABS + Q5 )*ZABS
     &            + Q4 )*ZABS + Q3 )*ZABS + Q2 )*ZABS + Q1 )*ZABS + Q0 )
*
*     |Z| &gt;= CUTOFF.
*
         ELSE
            P = EXPNTL/( ZABS + 1/( ZABS + 2/( ZABS + 3/( ZABS
     &           + 4/( ZABS + 0.65D0 ) ) ) ) )/ROOTPI
         END IF
      END IF
      IF ( Z .GT. 0 ) P = 1 - P
      PHID = P
      END
```

Appendix C. A Fortran Code to Compute Quantizer Threshold Spacing from Measured Zero-Lag Autocorrelation

This code is described in Section 4. Valid inputs are integer $n > 3$ and real zero-lag autocorrelation values $r(1)$ satisfying either $0 \le r(1) \le \left(\frac{n-1}{2}\right)^2$, for n odd; or $1 \le r(1) \le (n-1)^2$, for n even. v_1 is the first positive quantizer input threshold.

For the $n = 3$ case, $v_1 = \sqrt{2}\,\mathrm{erfc}^{-1}(r(1))$, where erfc^{-1} is the inverse of the error function. For this case, the inverse error function code of Appendix E could be used instead.

```
      double precision function thresh(n,zerolag)
      implicit double precision (a-h,o-z)
      pi=3.1415926535897932d0
      x=0d0
      if (mod(n,2).eq.0) x=1d0
      itmax=30
      tol=1d-8
      do i=1,itmax
         f=zerolag
         fp=0d0
         if (mod(n,2).eq.1) then
            do k=1,(n-1)/2
               f=f-(2*k-1)*erfc((2*k-1)*x/sqrt(2d0))
               fp=fp+sqrt(2d0/pi)*(2*k-1)**2*exp(-.5d0*((2*k-1)*x)**2)
            end do
         else
            f=f-1d0
            do k=1,(n-2)/2
               f=f-8*k*erfc(k*x/sqrt(2d0))
               fp=fp+8*k**2*sqrt(2d0/pi)*exp(-.5d0*(k*x)**2)
            end do
         end if
         deltax=-f/fp
         deltax=sign(1d0,deltax)*min(.5d0,abs(deltax))
         x=x+deltax
         if (mod(n,2).eq.1) x=max(0d0,x)
c        print *,i,x,deltax
         if (abs(deltax/x).lt.tol) go to 1
      end do
1     thresh=x
      return
      end
```

Appendix D. Mathematica Code to Derive 3-Level or 9-Level Quantizer Thresholds from Quantizer Zone Population Counts

```
(*  Given the population counts, n1, n2, and n3, in the quantization
level zones (corresponding to outputs -1, 0, and 1, respectively) of
a three-level quantizer, this procedure deduces the mean input level
as a fraction of the r.m.s. input level - assuming Gaussian statistics -
then deduces the input thresholds of the quantizer, and finally
constructs the effective  quantization function, for use within the
Van Vleck correction code.
    n=n1+n2+n3 is the total number of samples, p1 is the fractional
population within the -1 level zone, and p3 is the fractional
population within the +1 zone.  (p2=1-(p1+p3) then is the fraction
in the middle zone.)  muoversigma is the estimate of the population
mean, divided by the r.m.s.  2*v1 is the spacing of the quantizer
input thresholds.
    This function may be of use with the GBT correlator - which
computes the quantizer level populations for a brief period period
of time (20 msec, I believe) at the beginning of each integration -
if there is any "d.c. offset" in the correlator input.  It would
be preferable, though, if the population counts for the whole
integration period could be made available.
    The ideal value of the quantizer thresholds, for optimal efficiency
in the weak correlation limit, is v1=.612003 sigma.  For that case,
with no d.c. offset, the fractional populations are p1=p3=27.0268% and
p2=45.9464%.
    Normally, if there were no suspicion of d.c. offsets, one would
deduce the quantizer threshold values from the unnormalized zero-lag
autocorrelation measurement and not use this procedure at all.  *)

qf3[n1_,n2_,n3_]:=Module[{n,p1,p3,t1,t3,muoversigma,aoversigma,v1},
(* For error trapping, check that n1>0, n2>=0, and n3>0.  (If any is
      negative, the inputs are nonsense.  If n1=0 or n3=0, the inputs
      are not necessarily nonsense - i.e., these cases are possible,
      but the solution is indeterminate.)  I don't know how one would
      want to handle error trapping in glish.  *)
   n=n1+n2+n3;
   p1=N[n1/n];
   p3=N[n3/n];
   t1=InverseErf[1-2p1];
   t3=InverseErf[1-2p3];
   muoversigma=(t1-t3)/Sqrt[2];
   v1=aoversigma=(t1+t3)/Sqrt[2];
   {{-v1,v1}-muoversigma,{-1,0,1}}]



(*  This function, qf9, is like the one above (qf3), but for the
case of 9-level quantization.  The GBT correlator does not provide
occupancy counts for each of the nine quantization zones, but rather
only the counts in the lowermost three, middle three, and uppermost
three zones.
    Given these population counts, n1, n2, and n3, respectively,
this procedure works as described above.
```

```
    The ideal values of the quantizer thresholds for the 9-level case
are (-7v1,-5v1,-3v1,-v1,v1,3v1,5v1,7v1), with v1=.2669111 sigma.
For that case, with no d.c. offset, the fractional populations are
p1=p3=21.1643% and p2=57.6714%.  *)

qf9[n1_,n2_,n3_]:=Module[{n,p1,p3,t1,t3,muoversigma,aoversigma,v1},
(* For error trapping, check that n1>0, n2>=0, and n3>0.  (If any is
      negative, the inputs are nonsense.  If n1=0 or n3=0, the inputs
      are not necessarily nonsense - i.e., these cases are possible,
      but the solution is indeterminate.)  I don't know how one would
      want to handle error trapping in glish.  *)
   n=n1+n2+n3;
   p1=N[n1/n];
   p3=N[n3/n];
   t1=InverseErf[1-2p1];
   t3=InverseErf[1-2p3];
   muoversigma=(t1-t3)/Sqrt[2];
   aoversigma=(t1+t3)/Sqrt[2];
   v1=aoversigma/3;
   {{-7v1,-5v1,-3v1,-v1,v1,3v1,5v1,7v1}-muoversigma,
    {-4,-3,-2,-1,0,1,2,3,4}}]
```

Appendix E. A Mathematica Code for the Inverse of the Error Function

In this Appendix I present Mathematica code that can be used to approximate the inverse of the error function and the inverse of the complementary error function. Since (more accurate) approximations to these functions are already built in to Mathematica, I have named my versions `MyInverseErf` and `MyInverseErfc`. My purpose in writing this code was to provide templates appropriate for coding in Fortran or C, as would be needed by AIPS++. This code is based upon approximations published by Blair, Edwards, and Johnson [11].

The approximations given in [11] are best uniform rational approximations to erf^{-1} (i.e., "minimax", or so-called *Chebyshev* approximations), which are best in the sense that they minimize the maximum relative error over the given interval of approximation.

```
(* The following approximations to the inverse of the error function are
taken from J. M. Blair, C. A. Edwards, and J. H. Johnson, "Rational
Chebyshev Approximations for the Inverse of the Error Function",
Mathematics of Computation, 30 (1976) 827-830 + microfiche appendix. *)


(* This approximation, taken from Table 10 of Blair et al., is valid
for |x|<=0.75 and has a maximum relative error of 4.47 x 10^-8. *)
MyInverseErf[x_Real/;Abs[x]<=.75]:=Module[{t=x^2-.75^2,
    p={-13.0959967422,26.785225760,-9.289057635},
    q={-12.0749426297,30.960614529,-17.149977991,1.00000000}},
    x*(p[[1]]+t*(p[[2]]+t*p[[3]]))/(q[[1]]+t*(q[[2]]+t*(q[[3]]+t*q[[4]])))]

(* This approximation, taken from Table 29 of Blair et al., is valid
for .75<=|x|<=.9375 and has a maximum relative error of 4.17 x 10^-8. *)
MyInverseErf[x_Real/;Abs[x]>=.75&&Abs[x]<=.9375]:=Module[{t=x^2-.9375^2,
    p={-.12402565221,1.0688059574,-1.9594556078,.4230581357},
    q={-.08827697997,.8900743359,-2.1757031196,1.0000000000}},
    x*(p[[1]]+t*(p[[2]]+t*(p[[3]]+t*p[[4]])))/
      (q[[1]]+t*(q[[2]]+t*(q[[3]]+t*q[[4]])))]

(* This approximation, taken from Table 50 of Blair et al., is valid
for .9375<=|x|<=1-10^-100 and has a maximum relative error of 2.45 x 10^-8. *)
MyInverseErf[x_Real/;Abs[x]>=.9375&&Abs[x]<1.0]:=Module[{
    t=1/Sqrt[-Log[1-Abs[x]]],
    p={.1550470003116,1.382719649631,.690969348887,-1.128081391617,
       .680544246825,-.16444156791},
    q={.155024849822,1.385228141995,1.000000000000}},
    Sign[x]*(p[[1]]/t+p[[2]]+t*(p[[3]]+t*(p[[4]]+t*(p[[5]]+t*p[[6]]))))/
          (q[[1]]+t*(q[[2]]+t*(q[[3]])))]

(* Finally,  *)
MyInverseErf[-1.]=-$MaxMachineNumber
MyInverseErf[1.]=$MaxMachineNumber


(* Now, for the inverse of the complementary error function we'll
generally use the relation InverseErfc[x]=InverseErf[1-x] for larger
values of x, those satisfying .0625<=x<2 :   *)
MyInverseErfc[x_Real/;x>=.0625&&x<2.0]:=MyInverseErf[1-x]

(* But for smaller values of x we'll use the approximations from
```

```
Table 50 (again) of Blair et al., as well as Table 70. *)
MyInverseErfc[x_Real/;x>=10^-100&&x<.0625]:=Module[{t=1/Sqrt[-Log[x]],
    p={.1550470003116,1.382719649631,.690969348887,-1.128081391617,
       .680544246825,-.16444156791},
    q={.155024849822,1.385228141995,1.000000000000}},
    (p[[1]]/t+p[[2]]+t*(p[[3]]+t*(p[[4]]+t*(p[[5]]+t*p[[6]]))))/
          (q[[1]]+t*(q[[2]]+t*(q[[3]]))))]

(* This approximation, taken from Table 70 of Blair et al., is valid
for 10^-1000<=|x|<=10^-100 and has a maximum relative error of 2.45 x 10^-8. *)
MyInverseErfc[x_Real/;x>=10^-1000&&x<10^-100]:=Module[{t=1/Sqrt[-Log[x]],
    p={.00980456202915,.363667889171,.97302949837,-.5374947401},
    q={.00980451277802,.363699971544,1.000000000000}},
    (p[[1]]/t+p[[2]]+t*(p[[3]]+t*p[[4]]))/(q[[1]]+t*(q[[2]]+t*(q[[3]]))))]

(* Finally,  *)
MyInverseErfc[0.]=$MaxMachineNumber
MyInverseErfc[2.]=-$MaxMachineNumber
```